IN THE CLAIMS

Please amend claim 599 as indicated below. A complete listing of claims is presented below for the convenience of the Examiner.

1-505. (Cancelled)

506. (Previously Presented) A method for processing instructions in a central processing unit (CPU) capable of executing instructions of a plurality of instruction sets, including a stack-based and a register-based instruction set, the method, comprising:

maintaining data for register-based instructions from the register-based instruction set and an operand stack for operands associated with stack-based instructions from the stack-based instruction set in a first register file, wherein at least some of the operands are moved between the register file and memory via at least one of an overflow and underflow mechanism; Java™

maintaining an indication of a depth of the operand stack; and

executing the stack-based instructions and register-based instructions in an execution unit, including generating an exception in respect of selected stack-based instructions.

507. (Previously Presented) The method of claim 506, further comprising storing variables associated with the stack-based instructions in a second register file.

508. (Previously Presented) The method of claim 507, further comprising storing Java™ registers in a third register file.

509. (Previously Presented) The method of claim 508, wherein the first, second, and third register files are the same register file.

510. (Previously Presented) The method of claim 506, wherein the overflow mechanism generates an overflow indication for the stack-based operands.

511. (Previously Presented) The method of claim 506, wherein the underflow mechanism generates an underflow indication for the stack-based operands.

512. (Previously Presented) The method of claim 509, wherein the operand stack is maintained in a first portion of the register file, and variables associated with the stack-based instructions are maintained in a second portion of the register file.

513. (Previously Presented) The method of claim 506, further comprising generating a branch taken indication in respect of a selected stack-based branch instruction.

514. (Previously Presented) The method of claim 513, further comprising flushing at least part of a pipeline associated with the processing of the selected stack-based instructions if the branch taken indication is generated.

515. (Previously Presented) The method of claim 506, wherein a memory arbiter is used to facilitate at least one of a loading and a storing of operands between the register file and the memory via the at least one of the overflow and underflow mechanism.

516. (Previously Presented) The method of claim 506, wherein at least one of the operands are moved between the register file and the memory as a result of executing at least one of a store and load operation due to at least one of the overflow and underflow indication.

517. (Previously Presented) The method of claim 515 or claim 516, wherein the memory is a data cache.

518. (Previously Presented) The method of claim 513, wherein the selected stack-based branch instruction is selected from the group consisting of ifeq, ifne, iflt, ifge, ifgt, ifle, if_icmpeq, if_icmpne, if_icmplt, if_acmpge, if_cmpgt, if_icmple, if_acmpeq, if_acmpne, ifnull, ifnonull, lcmp, fcmpl, fcmpg, dcmpl, and dcmpg.

519. (Previously Presented) The method of claim 506, further comprising further processing the selected stack-based instructions for which exceptions were generated using the register-based instruction set.

520. (Previously Presented) The method of claim 519, wherein the further processing occurs within a virtual machine.

521. (Previously Presented) The method of claim 519 or claim 520, further comprising reverting to processing the stack-based instructions after the further processing.

522. (Previously Presented) The method of claim 506, wherein instructions of the stack-based instruction set include Java™ bytecodes.

523. (Previously Presented) The method of claim 516, wherein executing the load or the store operation is due to executing a load or a store instruction associated with the register-based instruction set.

524. (Previously Presented) The method of claim 506, wherein a common program counter register is used for the plurality of instruction sets.

525. (Previously Presented) The method of claim 506, wherein a program counter for each of the plurality of instruction sets is in at least one or more program counter registers.

526. (Previously Presented) The method of claim 506, wherein instructions for the plurality of instruction sets are stored in a shared instruction cache.

527. (Previously Presented) The method of claim 524 or claim 525, wherein the program counter register is part of a register file for the CPU.

528. (Previously Presented) The method of claim 506, wherein the CPU maintains an indication of which registers in the register file contain operands associated with the stack-based instructions.

529. (Previously Presented) The method of claim 528, wherein at least a top two operands of the operand stack in the register file are referenced when executing the stack-based instructions.

530. (Previously Presented) A method for processing instructions in a central processing unit (CPU), the method comprising:

decoding instructions of a first instruction set;

maintaining an operand stack associated with the instructions of the first instruction set in a register file including moving at least some operands between the register file and memory via at least one of an overflow and underflow mechanism;

decoding instructions of a second instruction set;

maintaining data associated with the instructions of the second instruction set in the register file;

sending an output of the decoding of the instructions of the first and second instruction set, to an execution unit; and

processing the output in the execution unit, including generating exceptions in respect of selected instructions of the first instruction set and processing the selected instructions for which exceptions were generated using a virtual machine.

531. (Previously Presented) The method of claim 530, further comprising setting at least one bit to indicate which instruction set to use for the processing.

532. (Previously Presented) The method of claim 531, wherein the processing is normally performed using the second instruction set.

533. (Previously Presented) The method of claim 531, wherein the at least one bit is set in respect of those instructions of the first instruction set for which an exception is generated.

534. (Previously Presented) The method of claim 532, wherein the first instruction set is a stack-based instruction set and the second instruction set is a register-based instruction set.

535. (Previously Presented) The method of claim 533, further comprising maintaining a program counter for the stack-based instruction set and a program counter for the register-based instruction set in the same register.

536. (Previously Presented) The method of claim 533, wherein the program counter for the plurality of instructions are in at least one or more registers.

537. (Previously Presented) The method of claim 530, wherein the output of decoding the first instructions is sent to the execute unit via the second decode unit.

538. (Previously Presented) The method of claim 530, wherein a memory arbiter is used to facilitate the loading and storing of operands between the register file and memory via the at least one of an overflow and underflow mechanism.

539. (Previously Presented) The method of claim 538, wherein the memory includes a data cache.

540. (Previously Presented) A method, comprising:

switching a processing system to an accelerator mode, wherein stack-based instructions are executed directly in hardware;

generating an exception in respect of a selected stack-based instruction while in the accelerator mode;

switching the processing system to a first native mode in which the selected stack-based instruction is processed within a virtual machine; and

switching the processing system to a second native mode upon a further exception, wherein in the second native mode the virtual machine is non-operative.

541. (Previously Presented) The method of claim 540, wherein the stack-based instructions include Java™ bytecodes.

542. (Previously Presented) The method of claim 540, wherein in the first and second native modes processing is performed using non-stack-based instructions.

543. (Previously Presented) A method, comprising:

processing instructions of a plurality of instruction sets in a CPU having an execution unit and a register file, wherein at least one of the plurality of instruction sets being a stack-based instruction set and at least one of the plurality of instruction sets being a register-based instruction set, using a common program counter for the plurality of instruction sets, the common program counter being stored in a common register; and

generating a branch taken signal to facilitate the processing of selected instructions of the stack-based and register-based instruction sets.

544. (Previously Presented) The method of claim 543, further comprising storing program counters associated with the processing in more than one register.

545. (Previously Presented) The method of claim 543 or claim 544, wherein the program counter is in a register file for the CPU.

546. (Previously Presented) The method of claim 543 or claim 544, wherein the instructions of the plurality of instruction sets are in a shared instruction cache.

547. (Previously Presented) The method of claim 543, wherein the selected instructions of the stack-based instructions are selected from the group consisting of ifeq, ifne, iflt, ifge, ifgt, ifle, if_icmpeq, if_icmpne, if_icmplt, if_acmpge, if_cmpgt, if_icmple, if_acmpeq, if_acmpne, ifnull, ifnonull, lcmp, fcmpl, fcmpg, dcmpl, and dcmpg.

548. (Previously Presented) The method of claim 543, further comprising flushing at least part of a pipeline associated when processing the selected instructions.

549. (Previously Presented) The method of claim 543, further comprising maintaining operands in an operand stack for the stack-based instructions in a register file for the CPU.

550. (Previously Presented) The method of claim 549, wherein the operands are moved between the register file and memory via at least one of an overflow and underflow mechanism.

551. (Previously Presented) In a processing system, comprising a central processing unit (CPU) having an execution unit and a register file capable of processing instructions of a plurality of instruction sets including a register-based instruction set and a stack-based instruction set, wherein an operand stack for the stack-based instruction set is maintained in the register file, operands are moved between the register file and memory due to at least one of an overflow and underflow mechanism, and wherein the processing system, further comprises a first state in which the CPU processes instructions using the register-based instruction set without a virtual machine, a second state in which the CPU processes using the non-stack-based instruction set within a virtual machine, and a third state in which the CPU processes instructions using the stack-based instruction set within the virtual machine, a method of operating the CPU comprising:

switching the processing system to the first state due to at least one of a reset command and a power-on condition;

switching the CPU to the second state;

processing instructions in the second state; and

upon encountering an exception while processing the instructions in the second state, switching the CPU to the first state.

552. (Previously Presented) The method of claim 551, further comprising switching the CPU from the second state to the third state.

553. (Previously Presented) The method of claim 552, further comprising, upon receiving a reset command, switching the CPU from the third state to the first state.

554. (Previously Presented) The method of claim 552, further comprising, upon encountering an exception while processing instructions in the third state, switching the CPU from the third state to the second state.

555. (Previously Presented) The method of claim 552, wherein the switching to the third state is while in a virtual machine to execute the stack-based instruction set.

556. (Previously Presented) The method of claim 551, further comprising setting at least one bit to indicate to the CPU which instruction set to use.

557. (Previously Presented) The method of claim 534, claim 540, claim 543 or claim 551, wherein the stack-based instructions include Java™ byte codes.

558. (Previously Presented) The method of claim 530, claim 540, or 551, wherein the virtual machine is a Java™ Virtual Machine.

559. (Previously Presented) A central processing unit (CPU), capable of executing a plurality of instruction sets comprising:

an execution unit and a register file to execute instructions of a plurality of instruction sets, including a stack-based and a register-based instruction set;

a mechanism to maintain at least some data for the plurality of instruction sets in the register file including maintaining an operand stack for the stack-based instructions in the register file and an indication of the depth of the operand stack;

a stack control mechanism that includes at least one of an overflow and underflow mechanism, wherein at least some of the operands are moved between the register file and memory; and

a mechanism to generate an exception in respect of selected stack-based instructions.

560. (Previously Presented) The central processing unit of claim 559, wherein the register file is a first register file, the central processing unit further comprising a second register file to store variables associated with the stack-based instructions.

561. (Previously Presented) The central processing unit of claim 560, further comprising a third register file to store Java™ registers.

562. (Previously Presented) The central processing unit of claim 561, wherein the first, the second, and the third register files are the same register file.

563. (Previously Presented) The central processing unit of claim 559, wherein the overflow mechanism generates an overflow indication for the operand stack.

564. (Previously Presented) The central processing unit of claim 563, wherein the underflow mechanism generates an underflow indication for the operand stack.

565. (Previously Presented) The central processing unit of claim 562, wherein the operand stack is maintained in a first portion of the register file, and variables associated with the stack-based instructions are maintained in a second portion of the register file.

566. (Previously Presented) The central processing unit of claim 559, further comprising a mechanism to generate a branch taken indication in respect of a selected stack-based instruction.

567. (Previously Presented) The central processing unit of claim 566, further comprising a mechanism to flush at least part of a pipeline associated with the processing of the selected stack-based instruction, if the branch taken instruction is generated.

568. (Previously Presented) The central processing unit of claim 559, further comprising a memory arbiter to facilitate at least one of a loading and a storing of operands between the register file and the memory and via the stack control mechanism.

569. (Previously Presented) The central processing unit of claim 559, wherein the operands are moved between the register file and the memory as a result of executing at least one of a load and a store operation due to at least one of the overflow and underflow indication.

570. (Previously Presented) The central processing unit of claim 559, wherein the memory is a data cache.

571. (Previously Presented) The central processing unit of claim 566, wherein the selected stack-based instruction is selected from the group consisting of ifeq, ifne, iflt, ifge, ifgt, ifle, if_icmpeq, if_icmpne, if_icmplt, if_acmpge, if_cmpgt, if_icmple, if_acmpeq, if_acmpne, ifnull, ifnonull, lcmp, fcmpl, fcmpg, dcmpl, and dcmpg.

572. (Previously Presented) The central processing unit of claim 559, further comprising further processing the selected stack-based instructions for which exceptions were generated using the register-based instruction set.

573. (Previously Presented) The central processing unit of claim 572, wherein the further processing occurs within a virtual machine.

574. (Previously Presented) The central processing unit of claim 573, wherein the execution unit reverts to processing the stack-based instructions, after the further processing.

575. (Previously Presented) The central processing unit of claim 559, wherein instructions of the stack-based instruction set includes Java™ bytecodes.

576. (Previously Presented) The central processing unit of claim 569, wherein executing at least one of the load and store operation is due to executing a load or a store instruction associated with the register-based instruction set.

577. (Previously Presented) The central processing unit of claim 559, further comprising a common program counter register for the plurality of instruction sets.

578. (Previously Presented) The central processing unit of claim 559, further comprising at least one program counter register to implement a program counter for each of the plurality of instruction sets.

579. (Previously Presented) The central processing unit of claim 559, wherein instructions for the plurality of instruction sets are stored in a shared instruction cache.

580. (Previously Presented) The central processing unit of claims 577 or claim 578, wherein at least some of the program counter is implemented within the register file.

581. (Previously Presented) The central processing unit of claim 559, further comprising a mechanism that maintains an indication of which registers in the register file contain operands associated with the stack-based instructions.

582. (Previously Presented) The central processing unit of claim 581, wherein at least a top two operands of the operand stack in the register file are referenced when executing the stack-based instructions.

583. (Previously Presented) A central processing unit (CPU) comprising:

a decoding mechanism to decode instructions of a plurality of instruction sets including a first instruction set and a second instruction set;

a register file, wherein an operand stack to store operands associated with instructions of the first instruction set is maintained; and wherein data associated with instructions of the second instruction set is maintained;

at least one of an overflow and underflow mechanism to cause the operands to move between the register file and memory; and

an execution unit that processes the output of the decoding of the instructions of the second instruction set, and the decoding of the instructions of the first instruction set, including

generating exceptions in respect of selected instructions of the first instruction set and processing the exceptions using a virtual machine.

584. (Previously Presented) The central processing unit of claim 583, further comprising a mechanism to set at least one bit to indicate which instruction set is to be used for the processing.

585. (Previously Presented) The central processing unit of claim 583, wherein an indication of the depth of the operand stack for the first instruction set is maintained.

586. (Previously Presented) The central processing unit of claim 584, wherein the at least one bit is set in respect of those instructions of the first instruction set for which an exception is generated.

587. (Previously Presented) The central processing unit of claim 583, wherein the first instruction set is a stack-based instruction set, and the second instruction set is a register-based instruction set.

588. (Previously Presented) The central processing unit of claim 584, further comprising a register within which a program counter for the stack-based instruction set and a program counter for the register-based instruction set is maintained.

589. (Previously Presented) The central processing unit of claim 583, wherein the decode unit comprises a first subunit and a second subunit, and wherein the first subunit decodes instructions

of the first instruction set and sends an output of the decoding to the execution unit via the second subunit.

590. (Previously Presented) The central processing unit of claim 583, further comprising a memory arbiter that facilitates at least one of the loading or storing of operands between the register file and memory via at least one of an overflow and underflow mechanism.

591. (Previously Presented) The central processing unit of claim 590, wherein the memory includes a data cache.

592. (Previously Presented) A processing system, comprising:

an accelerator mode in which a central processing unit (CPU) of the processing system processes stack-based instructions directly in hardware;

a first native mode in which the processing system processes using a non-stack-based instruction set within a virtual machine; and

a second native mode in which the processing system processes instructions using non-stack-based instructions, in which the virtual machine is non-operative, wherein

the processing system is switched to the accelerator mode to process stack-based instructions while in the accelerator mode, the processing of the stack-based instructions including generating an exception in respect of a selected stack-based instruction while in the accelerator mode, and switching to the first native mode in which the selected stack-based instruction for which the exception was generated is further processed within the virtual machine using the non-stack-based instruction set, and wherein if an exception is generated while in the first native mode, the processing system switches to the second native mode.

593. (Previously Presented) A central processing unit (CPU), comprising:

at least an execution unit and a register file to process instructions of a plurality of instruction sets, at least one of the plurality of instruction sets being a register-based instruction set, and at least one of the plurality of instruction sets being a stack-based instruction set, wherein at least one of an underflow and overflow mechanism is used to maintain the operand stack in the register file for the stack-based instruction set;

a register to store a common program counter for the plurality of instruction sets; and

a branch- taken indicator to facilitate the processing of selected instructions of the stack-based and register-based instruction sets.

594. (Previously Presented) The central processing unit of claim 593, further comprising a shared instruction cache to store instructions of the plurality of instruction sets.

595. (Previously Presented) The central processing unit of claim 593, wherein the selected instructions are selected from the group consisting of ifeq, ifne, iflt, ifge, ifgt, ifle, if_icmpeq, if_icmpne, if_icmplt, if_acmpge, if_cmpgt, if_icmple, if_acmpeq, if_acmpne, ifnull, ifnonull, lcmp, fcmpl, fcmpg, dcmpl, and dcmpg.

596. (Previously Presented) The central processing unit of claim 593, further comprising a pipeline, at least a part of which is flushed during processing of the selected instructions, due to a branch-taken instruction generated by the branch-taken indicator.

597. (Previously Presented) The central processing unit of claim 593, comprising a register file that implements an operand stack to store operands for the stack-based instruction set.

598. (Previously Presented) The central processing unit of claim 597, further comprising at least one of an overflow and underflow mechanism to move operands between the register file and memory.

599. (Currently amended) A processing system, comprising:

a central processing unit (CPU) includes an execution unit and a register file to process instructions of a plurality of instructions sets including a register-based instruction set and a stack-based instruction set;

a mechanism to maintain a stack for the stack-based instruction set in the register file with at least one of an underflow and overflow mechanism, wherein the processing system has a first state in which the CPU processes instructions using the register-based instruction set without a virtual machine, a second ~~stage~~ state in which the CPU processes instructions using the register-based instruction set within the virtual machine, and a third ~~stage~~ state in which the CPU processes instructions using the stack-based instruction set within the virtual machine, the processing system being configured to perform a method, comprising:

switching to the first state due to a reset command while in the third state or after power-on;

thereafter switching to the second state;

processing instructions while in the second state;

switching to the third state; and

upon receiving a reset command, switching from the third state to the first state.

600. (Previously Presented) The processing system of claim 599, wherein upon encountering an exception while in the third state, switching to the second state for further processing.

601. (Previously Presented) The processing system of claim 600, wherein upon encountering an exception while in the third state, the processing system switches to the second state for further processing of the exception.

602. (Previously Presented) The processing system of claim 600, wherein due to an exception while in the second state, the processing system switches from the second state to the first state.

603. (Previously Presented) The processing system of claim 600, wherein the processing system switches to the third state while in the virtual machine to execute the stack-based instruction set.

604. (Previously Presented) The processing system of claim 599, wherein the processing is done using register-based instructions for the first and second states.

605. (Previously Presented) The processing system of claim 604, wherein the processing system switches to the third state while in the virtual machine.

606. (Previously Presented) The processing system of claim 600 or claim 603, wherein an exception is generated for selected stack-based instructions.

607. (Previously Presented) The processing system of claim 599, wherein an error while in the third state switches the processing system to the second state.

608. (Previously Presented) The processing system of claim 603 or claim 605, wherein the virtual machine is a Java™ Virtual Machine.

609. (Previously Presented) The processing system of claim 603, wherein the stack-based instructions are Java™ bytecodes.